A Knowledge Engineering Methodology for Rapid Prototyping of Planning Applications

Luis Castillo and **Juan Fernández-Olivares** and **Antonio Gońzalez**
Gonzalo Milla and **David Prior** and **Lluvia Morales** and **José Figueroa**
Universidad de Granada

Víctor Pérez-Villar
Boeing Research and Technology Europe

Abstract

When deploying planning applications as part of greater intelligent systems there is an underlying knowledge acquisition, engineering and representation process that has to be undertaken. In this paper we propose a methodology that enables both a fast and easy knowledge acquisition, engineering an representation of planning knowledge and it also enables an easy knowledge sharing between the planning engine and either other subsystems or even end users.

Introduction

Knowledge acquisition and representation is a key issue in the development of planning applications either for small domains or for large-scale deployments. In most cases, planning engines are just one more part of the whole architecture and the integration of planning algorithms cannot ignore this because the domain knowledge that they use to work is also shared with other systems or even human operators. In addition to this, planning domain knowledge is not always explicitly stated since the beginning. The aim of this paper is enabling an easy *knowledge acquisition* stage of planning domain knowledge and enabling a good level of *knowledge sharing* either with other processes or human operators. In order to achieve this two-folded goal a methodology based on the use of well known technologies is proposed. We will focus on how, once the knowledge relative to the planning objects model and the planning problem has been modeled within Protégé Frames (National Library of Medicine 2009), this knowledge model can be easily and quickly translated into a representation understandable by a planner either an HTN planner or a flat one following a CommonKADS(Schreiber et al. 1999) methodology. When a brand new planning application is being built, the planning domain knowledge is under constant evolution and changes. The methodology presented in this paper allows both, an easier exchange of knowledge with other subsystems or human operators, and a quick and robust regeneration of most PDDL sentences both in the problem and in the domain files, leading to a faster and reliable evolution of the planning application.

The life-cycle of a brand new planning application

The building of a planning application from scratch is a handcrafted work nowadays. Planning engineers have to study the problem at hand and build a knowledge model. If some previous database is available it may help a little but since these data repositories were not designed to be used by a planning algorithm, they usually lack of the desired level of detail or precision and many times they need to be re-worked anyway. As the project goes on, and following a typical, spiral-based knowledge acquisition scheme based on interviews and prototypes, new capabilities of the planning engine are incrementally agreed between end users and planning engineers. This leads to an enormous amount of planning engineering work in the form of representing new PDDL predicates, initial states, objects, properties and so on. And last, but no least, potential syntactic mistakes may appear during PDDL hand-coding.

The CommonKADS underlying timeline

Planning engineers may follow the CommonKADS methodology to give a formal envelope to this knowledge engineering effort. Previous works (Kingston, Shadbolt, and Tate 1996) show that, since a planning application is also a knowledge-based system, this methodology can be used as the underlying timeline for the design and representation of planning domain knowledge. In a planning setting, the *Agent* and *Organization models* might gather all the knowledge available about the domain objects that take part in the system, the relations between them and the existing constraints, that finally is represented as the *domain knowledge* category of the *Knowledge model*. The *Task model* may be used to detect those knowledge-intensive processes that might be better supported by planning techniques. The Knowledge model (apart from the object model) may contain all inferencing mechanisms of the problem, that is, abductive or deductive inference processes needed in the planning process or even the knowledge relative to actions and to compound tasks, decomposition schemes, in the case of HTN planning. *Communication model* may be used to establish a way to define mixed-initiative or collaborative problem solving strategies, though it does seem to provide much better support in a planning application for the stages of plan

execution and monitoring, that is out of the scope of this paper. Finally, the *Design model* might be understood as the representation of all the above knowledge sources into the final representation, in our case, the PDDL domain and problem files ready to be used by a planning engine. Let us see how the ontology editor Protégé may be used to cover the agent, organization and knowledge models.

The use of the Protégé ontology editor

Firstly, it allows to easily represent planning domain knowledge (except that related to actions) in a well known framework with a graphical interface, focusing on the relevant aspects of this knowledge: objects, properties and relations and ignoring the details of such representation in predicate logic. The knowledge represented in Protégé Frames form can be translated into usual PDDL syntax with an automated procedure and all the sections of PDDL 2.1 domain and problem file can be automatically generated. This translation process is a great advantage since it immediately provides valid PDDL files after each revision of the ontology. Secondly, the knowledge in the Protégé ontology can also be easily shared and exchanged both with other components of the application like web applications, GIS systems like Google Maps, or other databases and also with human operators. This allows end users to modify planning domain knowledge from easy-to-use web forms without having to know about ontologies or predicate logic or artificial intelligence in general. In order to provide these advantages, the following steps have been given. On the one hand, the Protégé ontology has to be backed up on a back-end database in MySQL. This is automatically done through the Protégé GUI. On the other hand, a web service named Ontoserver has been designed and implemented over that back-end providing simultaneous access through the web to the ontology and allowing most kind of queries to it, such as browsing the hierarchy of classes, querying instances, following links of properties, etc.

Automating planning domain and problem file generation during the planning application life-cycle

Thanks to the use of the Ontoserver service, the ontology can be fully browsed and all the available knowledge can be easily translated into PDDL form. All the sections of a PDDL domain file are generated, except those related to actions schemes. One of the easiest PDDL sections to be generated is the :TYPES section. It is generated just by browsing the hierarchy of available classes in the ontology. Each class in the ontology is translated into a valid PDDL type. Another very useful section is the :CONSTANTS section. This section must contain all valid symbols to be shared between any problem instance of the planning domain. In this case, all the possible values of a given enumerated slot are translated into valid constants. The PDDL section :PREDICATES and :FUNCTIONS are extremely important since they are the main tool to encode the most important knowledge to be considered by PDDL actions and states. Every instance of the ontology has a set of slots. Every slot represents a property (STRING, SYMBOL, INTEGER, BOOLEAN, FLOAT) and has a value associated to it. Non-numerical slots are treated as regular predicates following the scheme `(<slot-name> <instance-name> <slot-value>)`. BOOLEAN slots a treated in a special way following the closed world assumption, if they are true, then they are translated as `(<slot-name> <instance-name>)`. Otherwise they are not translated. Numerical slots are translated as functions following the scheme `(= (<slot-name> <instance-name>) <numeric-value>)`. And finally, slots of type INSTANCE, that represent binary relations between instances (or classes), are translated following the scheme `(<slot-name> <source-instance-name> <target-instance-name>)`. It is very easy to figure out that the problem file follows the same scheme of the browse-and-translate procedures seen before. Every instance of the ontology is translated into a PDDL typed object in the problem. The type of the object is the class of the ontology it belongs to. And the initial state is a simple dump of all the slots of all the instances present in the ontology. The goal of the problem can be specified as a parameter of the translation algorithm. In the case of non-HTN planners, it must consist of a sequence of literals to be made true. In the case of HTN planners it must consist of a sequence of tasks to be decomposed.

Regarding knowledge sharing, having both plain PDDL domain and problem files as the source files for the planning knowledge is a huge drawback to enable this exchange of knowledge between all the agents involved in the whole problem. However, given the infrastructure commented in previous sections, this task is much easier. It's time to recall that Ontoserver is a web service that allows to browse the ontology, its classes and instances through a clearly defined API (Advanced Programming Interface). This allows other systems to remotely access the ontology in a interoperable manner and also to build interfaces so that human operators can also access the ontology through any web browser.

Acknowledgements

This research is partially funded by the Spanish Center for the Development of Industrial Technology (CDTI) through the CENIT Program, part of the 2010 Ingenio initiative. This support is gratefully acknowledged.

References

Kingston, J.; Shadbolt, N.; and Tate, A. 1996. CommonKADS models for knowledge based planning. "University of Edinburgh Artificial intelligence Applications Institute, Technical Report".

National Library of Medicine. 2009. http://protege.stanford.edu/. Version 3.4.1.

Schreiber, G.; Akkermans, H.; Anjewierden, A.; de Hoog, R.; Shadbolt, N.; de Velde, W. V.; and Wielinga, B. 1999. *Knowledge Engineering and Management – The CommonKADS Methodology*. The MIT Press.